Report on K-nearest neighbor algorithm with Radius and KNN with iterating radius.

**Radius KNN**

* -Import the data from the given text file in form of a Matrix.
* -Add 2 columns in the matrix. One to note the distances and other to give estimated class.
* -Iterate the below steps for the increasing values of radius.
* -Shuffle the matrix vertically randomly and slice it into a test set and training set. Now for each of the point in the test set, calculate the Euclidean distance with each of the point in the training set.
* -Sort the training matrix in the ascending order of the distances. Check what all points are in the specified radius.
* -calculate the points belonging to class 1 and class 2 and whichever is more in quantity, assign the current datapoint that specific class.
* -From Fig.1,2,3,4,5 , we could see that most of the performance metrics are high for the radius=0.8.
* -So to draw the decision boundary, we take the radius =0.8 and for each point on the meshgrid, we calculate the neighbors around it in that radius. Calculate how many are from class 1 and from class 2 and assign a value to that meshgrid point pertaining of that of max of class1 or class2.
* -We have given +1,-1,0 for all the points on meshgrid based on class 2 majority, no neighbors, class 1 majority. We have decided to take -1 into consideration because, if we don’t do that, approximation will be huge.

**KNN with multiple independent iterations.**

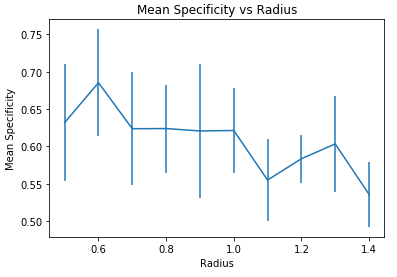
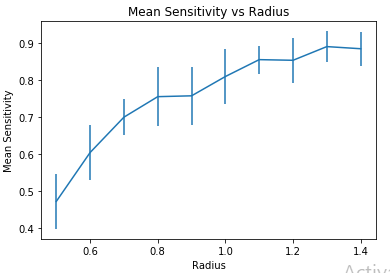
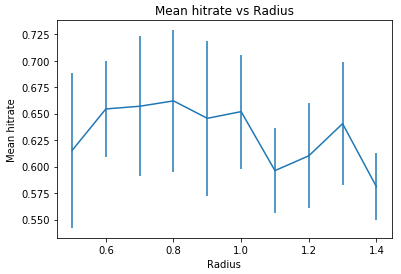
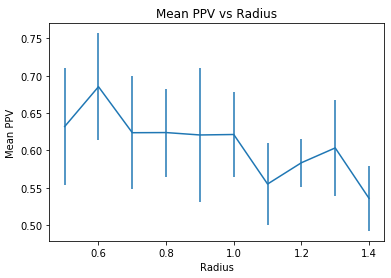
* Same data strategy followed in this as above, just instead of checking with all the neighbors in a certain proximity, we are checking with a certain prefixed number of closest neighbors.
* We add an augmentation here to check for the classwise mean distance between neighbors of different classes and assign value of class whose mean distance comes out to be minimum.
* We could see this by an example as follows. Let say if we are considering 5 neighbors. A as our datapoint has 5 neighbors B, C, D, E, F out of which B, C belong to class 1 and D, E, F belong to class 2. If we calculate the distance as (dist(B)+dist(C)) and (dist(D)+dist(E)+dist(F)), probability is that we will get the later value as bigger that the prior even though if D, E, F are individually small and thus closer to A.
* Thus, we consider to take the mean value ((dist(B)+dist(C))/2) and ((dist(D)+dist(E)+dist(F))/3) while making our conclusion about the classification.
* Now while we have selected our augmentation, we select the optimal neighbors as 2 in our hypothesis because, the performance metrics in Fig. 6,7,8,9 and 10 show us that these are best for neighbors=2

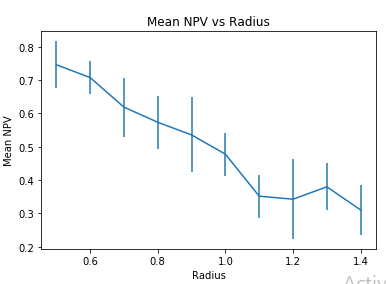
**Review between 2 above algorithms:**

* First thing to note here from decision boundary is that, the areas of the meshgrid which are far away from datapoints (white space in fig.11), can be well classified by the KNN but the radius KNN fails in such cases and depends hugely on approximation.
* Moreover, all the performance metrics for KNN are better than KNN radius for optimal value of radius =0.8 and neighbors=2.
* The KNN radius algorithm will consider that the data is in circular pattern around the datapoint which is hardly the case in real datasets. So its safe to say that in this case KNN with augmented classwise mean distance will perform better.

**Graphs for Mean of performance over independent repetitions vs the increasing Radius selected in the algorithm. Also, the error bars of the deviation.**

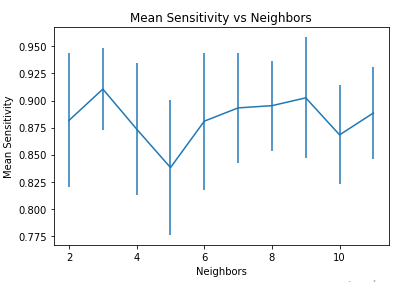
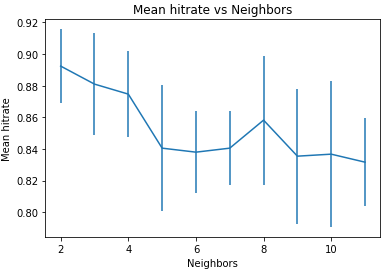
**Fig.1 ,2,3,4,5**

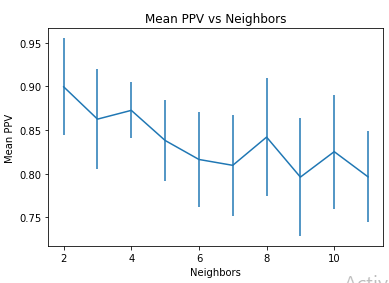
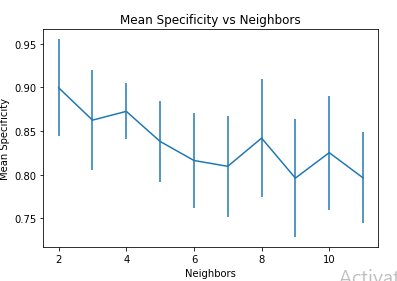


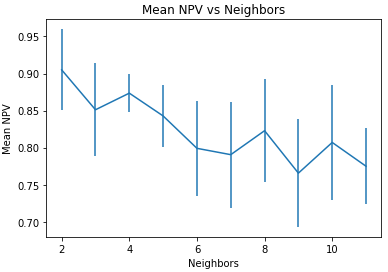


**Graphs for Mean of performance over independent repetitions vs the increasing numbers of neighbors in the algorithm. Also, the error bars of the deviation.**

Fig.6,7,8,9,10

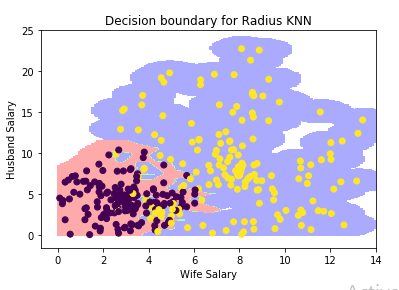






**Fig.11 Decision Boundary for the KNN classifier with Radius as the parameter for number of neighbors.**

**Red=Not stressed, Blue= Stressed, White =Cannot be classified by the classifier and too much approximation to give a default value.**



**Fig.12. Decision Boundary for the KNN classifier with classwise mean distance augmentation for neighbors=2**

**Red=Not stressed, Blue= Stressed**

